Project Lab Sheet B: Stepper Motor Control

Two projects must be submitted, one no later than week 11; the other no later than week 12.

1 Aims

The aim of this lab is to implement a steeper motor control, using the Programmable Interrupt Timer (PIT) to control the stepping speed precisely,

1.1 Example Projects

The following example projects are available. You may adapt code from these projects to complete the requirements described here.

1. **Steeper motor**: this gives information on the use of the stepper motor and includes an API to initialise and control it.
2. **Tone Generator**: the frequency control project shows how to use the PIT and provides a small API for this.

2 Project Requirements

The requirements of the system are described below.

1. The start position of the motor is its position when the program starts.
2. Two buttons control the stepper motor.
3. The first button, the 'run' button, causes the motor to move, provided that the motor is in the start position. The system has a number of moves, shown in the table below, in the time shown. On each press of the button, the motor advances to the next move, before returning to move 1.
4. A second button, the 'stop button', behaves as follows:
   - If the motor is moving, pressing the stop button causes it to stop.
   - If the motor is not moving, for example because the move has been completed, then pressing the button causes the motor to return to its start position, using the minimum number of steps.
5. The stepper speed must be controlled by the PIT.

The stepper must be able to make the following 8 moves:

Move Number	Number of Turns	Time Taken (sec)	Direction of Rotation
1	1 1/3	20	Clockwise
2	5 2/3	20	Clockwise
3	10	20	Anticlockwise
4	20 1/3	20	Anticlockwise
5	10 2/3	10	Anticlockwise
6	20	10	Anticlockwise
7	30 1/3	10	Clockwise
8	40 2/3	10	Clockwise

For example, one scenario would be:

- The move button is pressed and the motor begins move 1. We can call the starting position 12 o'clock.
- After 20 seconds exactly, having move 64 steps clockwise (as 48 + 16 = 64), the motor stops at 4 o'clock.
- The stop button is pressed and the motor rotates anticlockwise 16 steps to return to 12 o'clock.

You will need to clarify these requirements to make the behaviour precise. You can do so in any way you choose[1].

3 Suggestions

This section contains some additional information that may be useful. You do not have to read this section if you do not wish to.

3.1 Software Requirements

Work out the detailed software requirements. The scenario above is simple but others that are more complex. For example:

- If the stop button is pressed before the move is complete, you will need to calculate the quickest move back to the start position.
- The stop button could be pressed as the motor is returning to its start position. Since the motor is moving, it is clear that it should stop.

Other requirements are not clear. For example:

- Is it always necessary to press the stop button between moves?

It is best to work incrementally: start simply and then add more functionality. You may also wish to set the on-board LEDs as a simple way to confirm the operation is an expected.

3.2 Use of Pins

Create a table of the I/O pins by function, pin name and FRM-KL25Z header/pin number. You need this to wire the system correctly. In addition, it is possible that the pins used in the demonstration projects clash (*if so, it is not deliberate*).

3.3 Software Design

For a cyclic system[2] design:

- What tasks will you have?
- What states does each task (or the system overall) have?
- How the tasks communicate?

A small part of the Stepper control code from the demo program needs to be called from the Timer ISR. You do not need to modify the code extensively.

3.4 State Transition Diagram

You are strongly recommended to draw a state transition diagram. The best tool to use for this is a pencil.

[1] You are demonstrating your ability as a developer: do not overelaborate the system but very simple solutions may gain slightly fewer marks.

[2] You can use the RTOS if you wish.

4 Answer Sheet

This sheet should be printed out and handed in during the lab session. It can be completed either electronically or by hand.

Surname		**First name**	
Student number			

4.1 Information About Your System

4.1.1 Requirements

Explain exactly what your proposed system does. Note any requirements that are not implemented.

4.1.2 Design

Detail the main elements of the design of your system, including the cycle time, the tasks, the communication between tasks and the states of tasks. You may wish to draw one or more state diagram.

4.2 Viva Record

You should demonstrate your system, at least once. You may choose to do more demonstrations if you are working incrementally.

Initials / Date	Comments Describing Functionality Observed

4.3 Overall Feedback

Marker	**Date**	**Grade**